04

Excepciones y errores

Transcripción

[00:00] Ya tenemos aquí el diagrama nuevamente y ya estamos viendo aquí que la excepción que yo he creado, ya está aquí. MiException extiende de RuntimeException, que a la vez se extiende de Exception y a la vez extiende de Throwable. Suena como un trabalenguas, pero en un momento ya va a tener un poco más de sentido.

[00:23] Y la duda que a mí me quedó en el video anterior fue por qué es que yo no consigo, pues digamos extender directamente de Throwable, si al final ya vimos en el código que Throwable es quien hace todo el trabajo, digamos. Él es quien imprime el stackTrace, él es quién define qué va a hacer con los parámetros que le enviamos al constructor, él es quien hace todo ese trabajo. Bueno.

[00:52] La explicación es un poco simple. En Java tenemos dos grandes grupos que extienden de Throwable. Tenemos, por un lado, las excepciones, exceptions. Y por otro lado tenemos los errores. ¿Cuál es la diferencia entre estos dos? Las excepciones como tal son los errores que son lanzados por nosotros cuando estamos programando normalmente cuando estamos, digamos, creando clases, todo eso usando la JVM.

[01:30] Cada cosa que nosotros hagamos sobre la JVM va a lanzar excepciones, que son los tipos de errores que nosotros podemos controlar, que el programador puede controlar. ¿Y entonces qué es un error? El error viene a ser el tipo de error lanzado por la JVM. Entonces vamos a hacer una diferencia aquí.

[01:54] Esta es la JVM, lo que está encima de la JVM son excepciones y lo que está dentro de la JVM son errores. Esos errores son los que lanzan los que programan la JVM. La gente de Oracle que da mantenimiento a la JVM no lanza excepciones, lanza errores. ¿Por qué?

[02:17] Porque recordemos que la JVM lo que hace ya es trabajar con bytecode a un nivel ya de memoria directamente. Entonces ustedes aquí pueden ver que yo cité un error muy conocido, muy común, que es el StackOverflowError. Nos damos cuenta en la nomenclatura que cuando nombramos excepciones, llega, digamos el nombre de la excepción más el complemento exception.

[02:46] En el caso de los errores, es la misma regla, nombre del error y el complemento, error, para diferenciar justamente eso. Entonces, y solo para resumir aquí, excepciones es el tipo de errores que retornamos los que programamos sobre la JVM, nosotros los programadores normales. Bueno, no normales, pero los que hacemos features basándonos en la JVM.

[03:12] Y los errores son el tipo de errores retornados por los que programan la JVM en sí. Y para dar un mejor ejemplo de lo que es un error yo aquí voy a generar un error y justamente este error, el StackOverflowError. ¿Cómo lo voy a hacer? Muy sencillo, el StackOverflowError es un error que se da cuando la memoria y cuando el stack propio dicho, cuando la pila se llena.

[03:440] Ustedes dirán, ¿pero cómo se puede llenar la pila? Simplemente yo puedo hacer eso aquí. Método 2. ¿Por qué? Porque yo tengo aquí un método 2 que llama a método 2. Ahí yo ya tengo un ciclo infinito, porque método 2 llega aquí y él va a llamar a método 2, y él va a llegar aquí y él va a llegar acá. ¿Entonces, qué va a pasar con mi stack?

[04:11] Si yo tenía main método 1, método 2, método 2, va a llamar a método 2, llega aquí, método 2 va a llamar a método 2, llega aquí, método 2 va a llamar a método 2 y así va a seguir hasta el infinito, y todo tiene un límite, entonces

vamos a ejecutar esto. Guardamos y listo. Vemos que aquí dio error en algún punto.

[04:34] Si nosotros subimos, subimos, subimos todo el todo el flujo, vamos a ver que él dio aquí StackOverflowError, excepción en el hilo main. ¿Dónde? En flujo Java línea 23. Aquí, método 2, entonces este error no puede ser controlado por nosotros. ¿Por qué? Porque si es que simplemente ya no hay más memoria en el stack no hay forma ninguna que nosotros consigamos hacer algo al respecto.

[05:02] Simplemente se acabó la memoria. Muy diferente al caso de la excepción de aquí, donde si detona una bomba del tipo ArithmeticException porque ya sea dividiste entre 0 o NullPointerException porque la variable hacía referencia a un campo nulo en la memoria, son cosas de las que el sistema se puede recuperar de cierta forma.

[05:27] Yo puedo adicionar cierta lógica para, en caso de que retorne null, NullPointerException, entonces crea un nuevo objeto y sigue el camino. Solucionado. En el caso de los errores no es así. El error simplemente ocasiona que si mi máquina se quedó sin memoria, automáticamente mi programa va a morir, no hay nada más que hacer, no hay más ciencia ahí, no hay forma que nosotros consigamos controlar ese error.

[05:51] Entonces esa es la diferencia básica entre estos dos grandes grupos de excepciones y los errores. Entonces esa es, digamos, la primera razón por la cual yo no puedo extender MiException de throwable, porque yo estaría quebrando la regla de Java de tener solamente dos grupos de errores, errores como tal y excepciones, yo no puedo hacer eso.

[06:21] Sería un error mío si yo extendía directamente de throwable. Y ahora bajamos un nivel. ¿Será que puedo extender de Exception, como está aquí? ¿Será que consigo hacer eso? Vamos a descubrirlo en el siguiente video.